2023-2 Capstone Design Project

자리 있나요?
Empty Seats?

Final Presentation

Team A (A's) 김도엽 박재윤 우다연 최지민

Contents

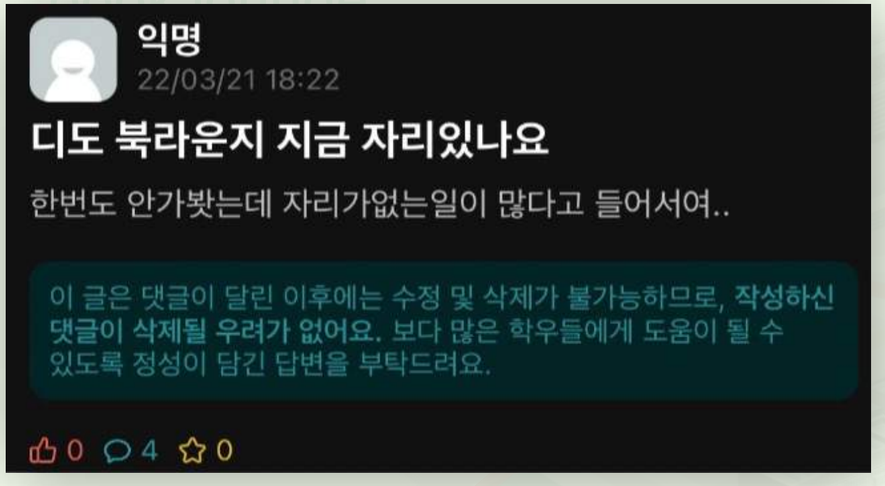
Study Spaces

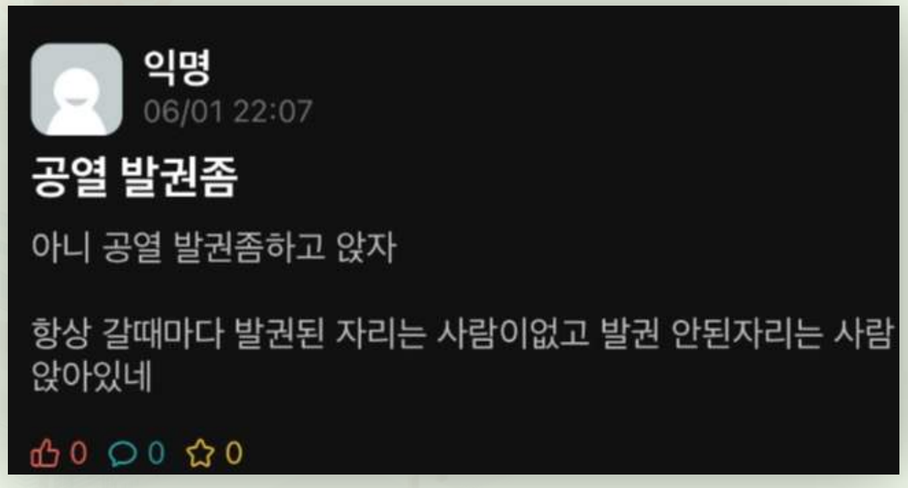
Samsung library
book lounge

Haedong
library

Eskara lounge

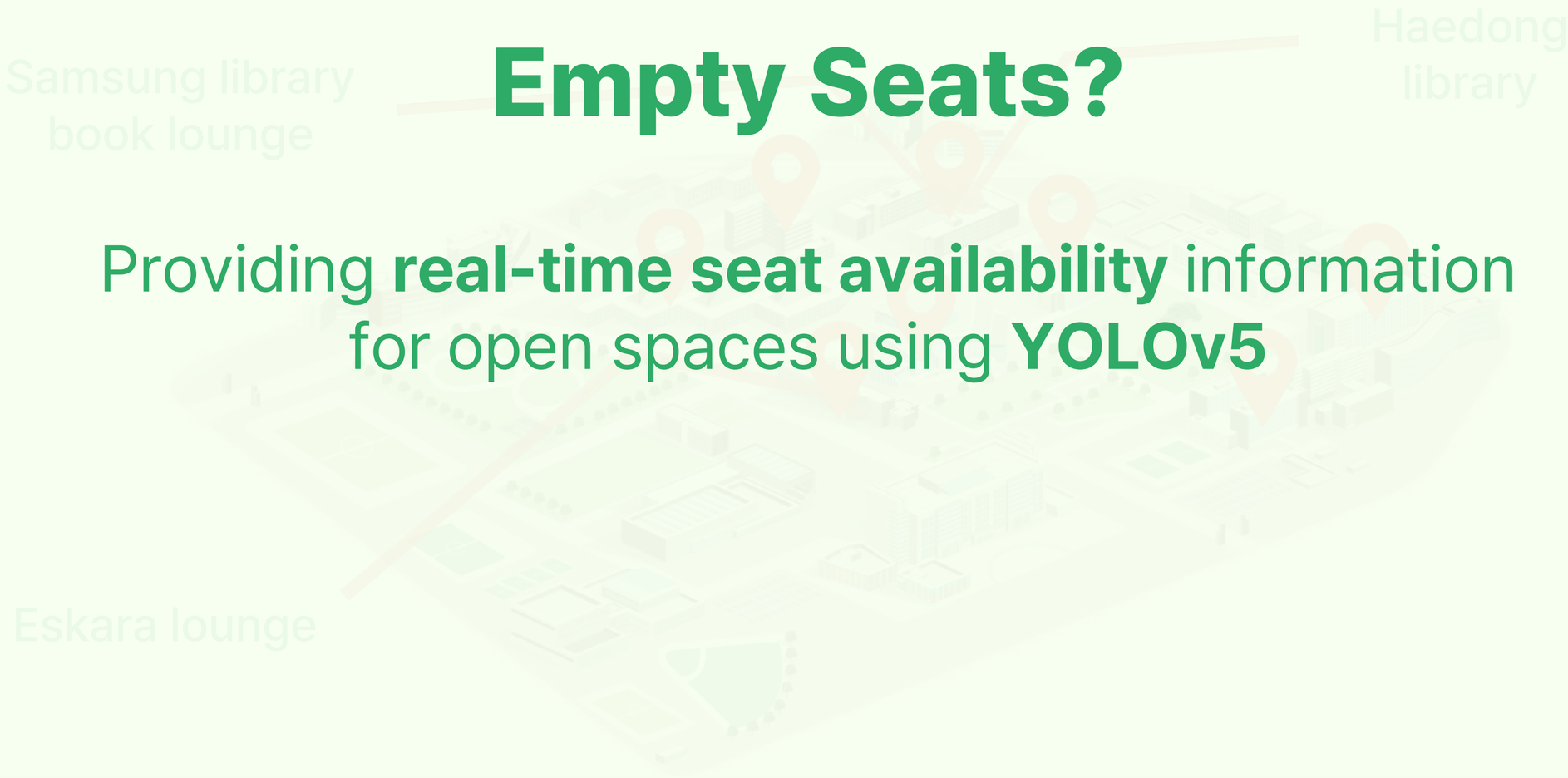
Checking for **empty seats** without **visiting** is challenging





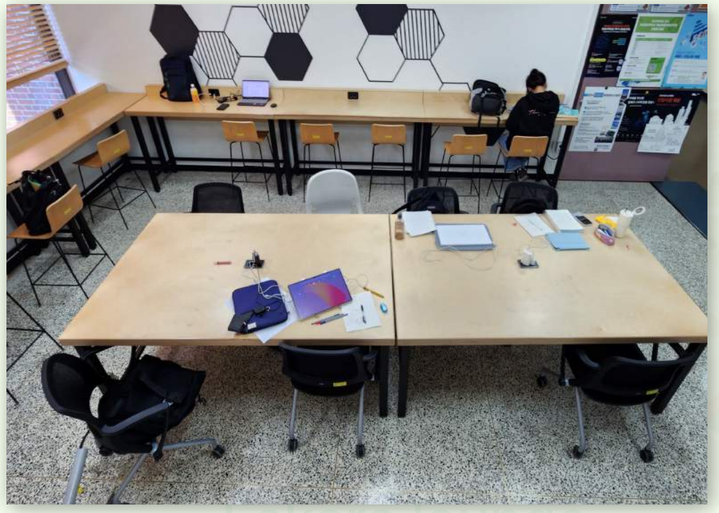
Empty Seats?

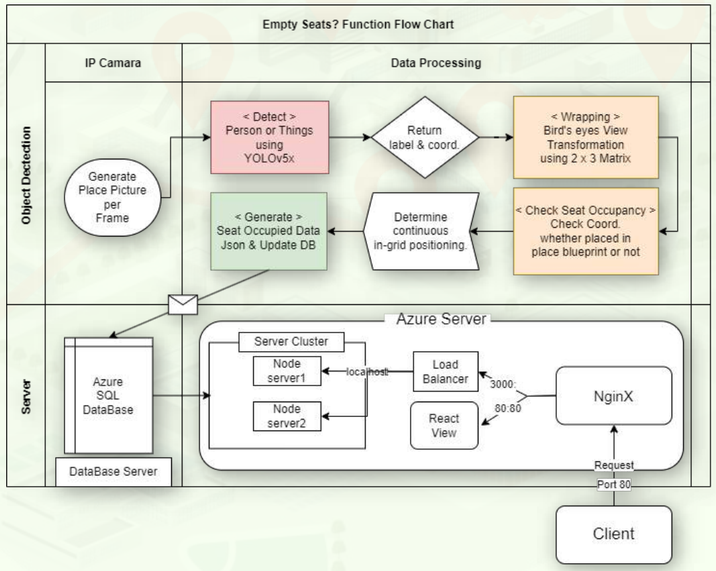
Providing **real-time seat availability** information
for open spaces using **YOLOv5**

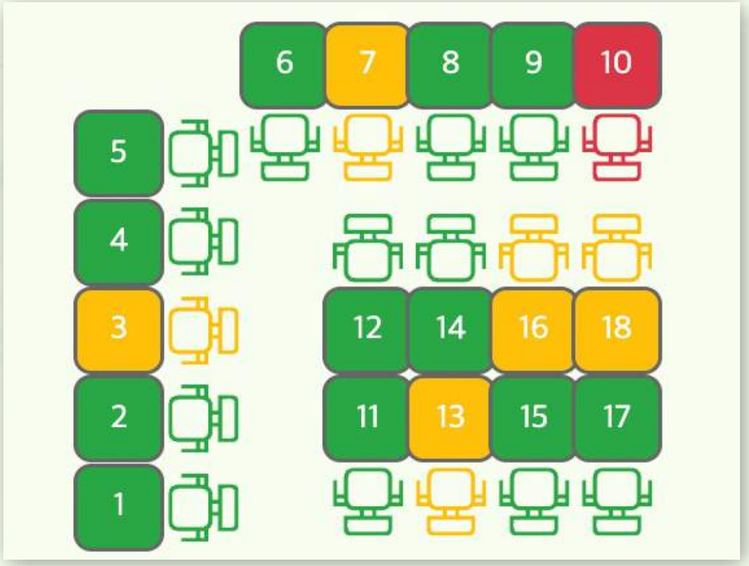


Empty Seats?

Input → **Processing** → **Output**







2. Project Progress

Roles

최지민

- YOLO management
- Frontend

김도엽

- **Team leader**
- Backend manager
- Code integration

박재윤

- Frontend manager
- Util features management

우다연

- Clerk & PPT
- AI & Frontend Development

PART 02 I Project Progress

Schedule

회의록

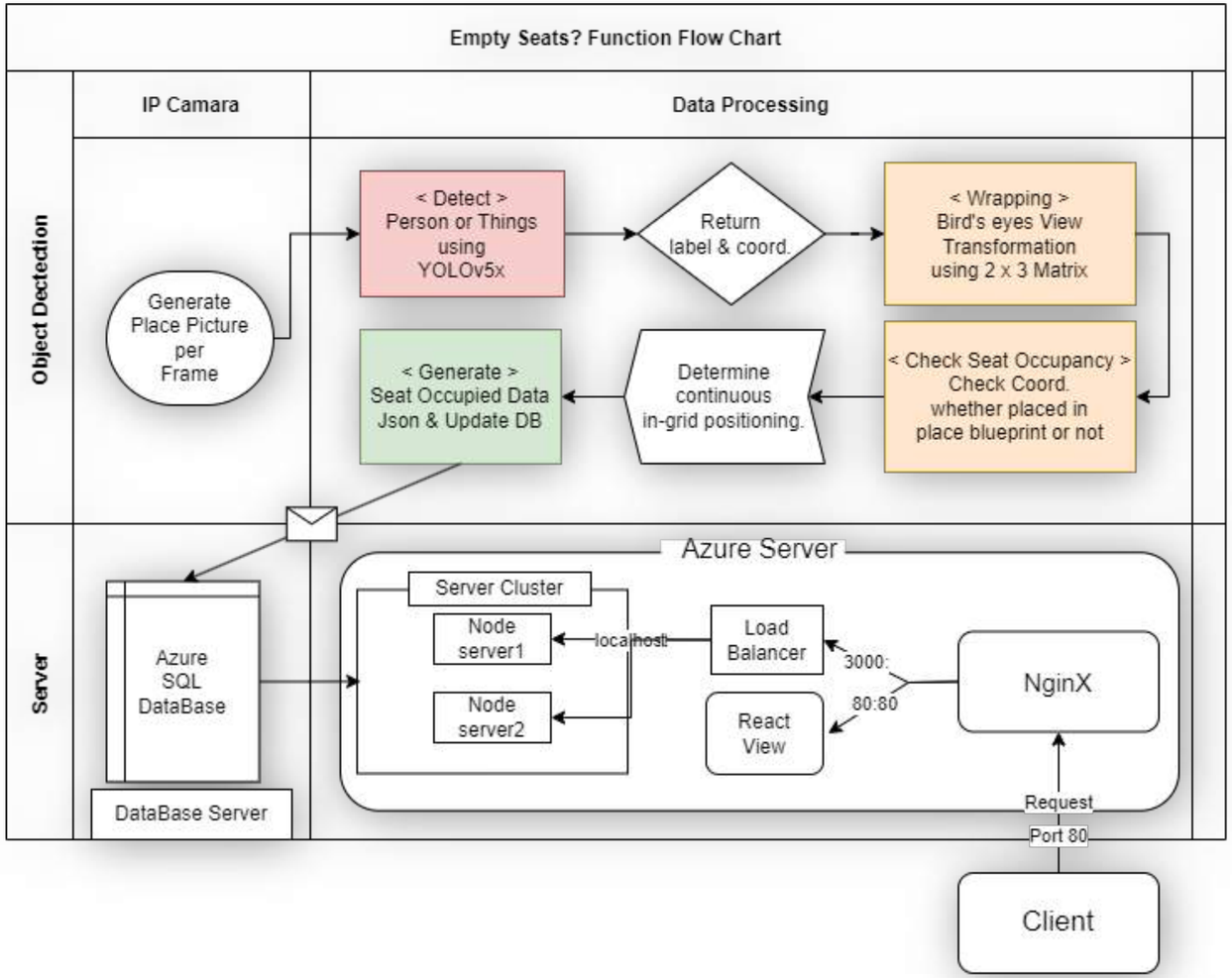
Aa Name	Date	회의 종류
0. 팀 빌딩 & 주제 방향성 논의	2023/09/08	
주제 회의	2023/09/09	
주제 구체화 회의	2023/09/10	
모델 구현 아이디어 회의	2023/09/22	금요일 대면
역할 분담 및 세부 일정 조율	2023/09/25	월요일 줌
To Do & Done	2023/09/26	추가회의 대면
Weekly Progress (1) 발표 준비	2023/10/04	추가회의 대면
실제 구현 회의 1	2023/10/06	금요일 대면
실제 구현 회의 2	2023/10/09	월요일 줌
실제 구현 회의 3	2023/10/13	금요일 대면
Weekly Progress (2) 발표 준비	2023/10/13	목요일 대면
Midterm 발표 준비		
실제 구현 회의 4	2023/10/29 11:00 PM	줌 추가회의
실제 구현 회의 5	2023/10/30 11:00 PM	줌
Front 구현 회의 1	2023/11/14	줌 추가회의
최종 구현 회의 1	2023/11/17	금요일 대면
최종 구현 회의 2	2023/11/20	월요일 줌
최종 회의 1	2023/12/01	줌 금요일
최종 회의 2	2023/12/04	대면 월요일
최종 회의 3		

	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11	Week 12	Week 13	Week 14	Week 15
Refinement of the Topic														
AI Model Design														
Data Preprocessing														
Model & Algorithm Implementation														
UI/UX Design														
UI/UX Implementation														
Mechanism Integration														
Beta Service Launch														

3. Implementation

PART 03 I Implementation

Architecture



IP Camera

↓

Data Processing

↓

Azure SQL Database

↓

Azure Server

↓

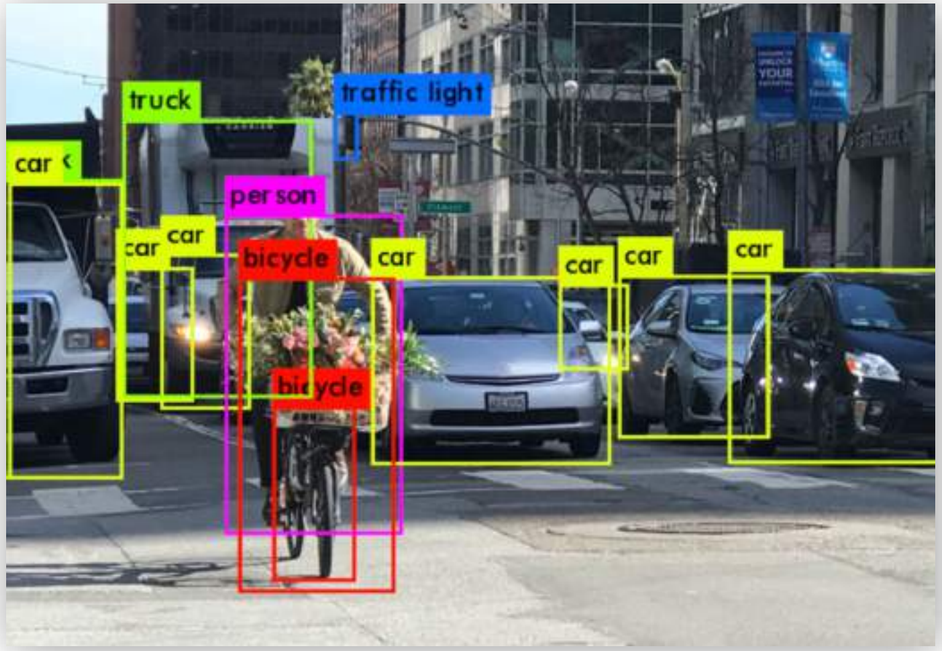
Client

IP Camera

- IP cameras stream videos over the network via Real-Time Streaming Protocol(RTSP).
- The RTSP video capture feature supplies the essential image source for YOLOv5 Object Detection.

Background

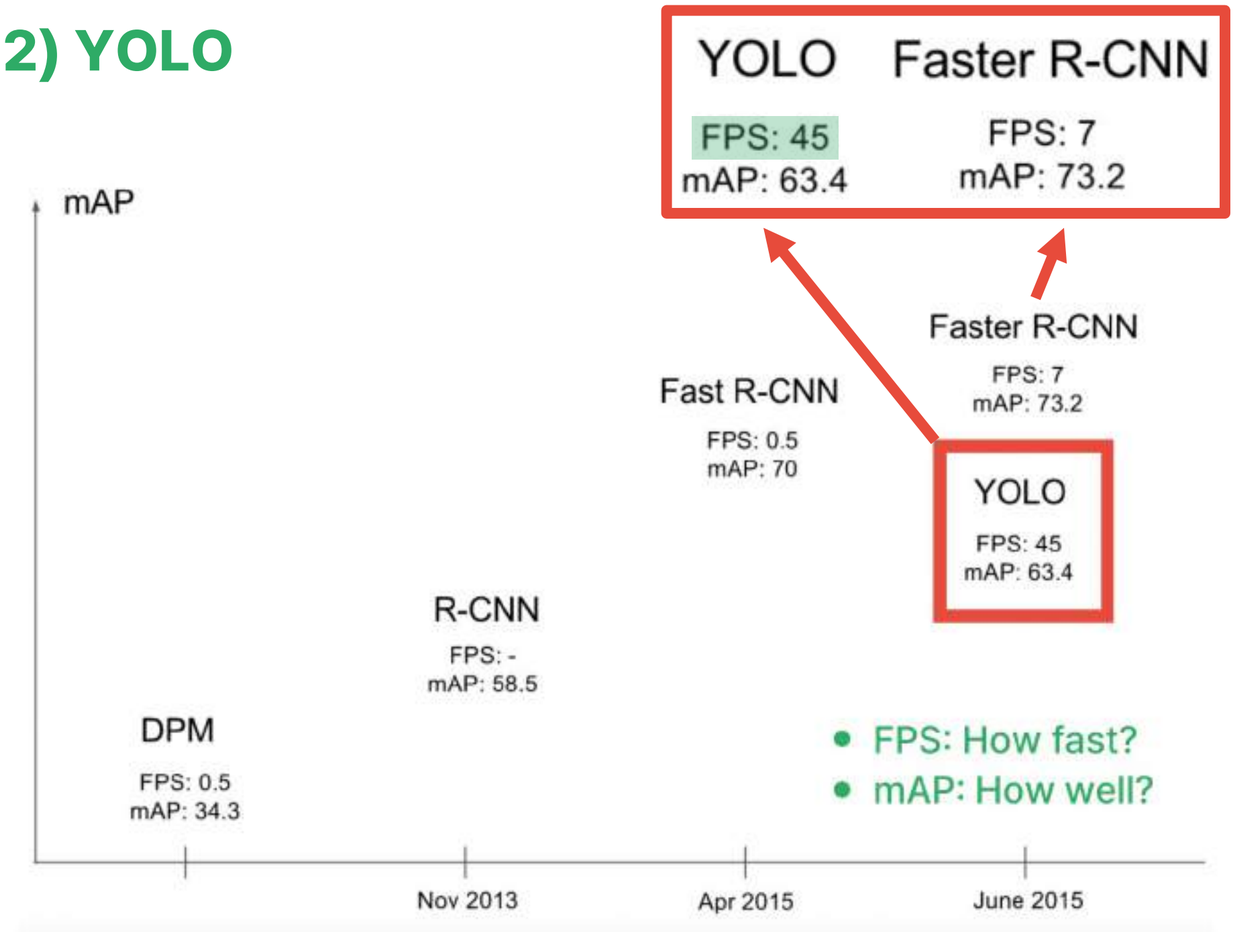
1) Object Detection



A computer vision task that involves **identifying** and **locating** objects

Background

2) YOLO



YOLO > Other models

- Much **faster**
- More **accurate**

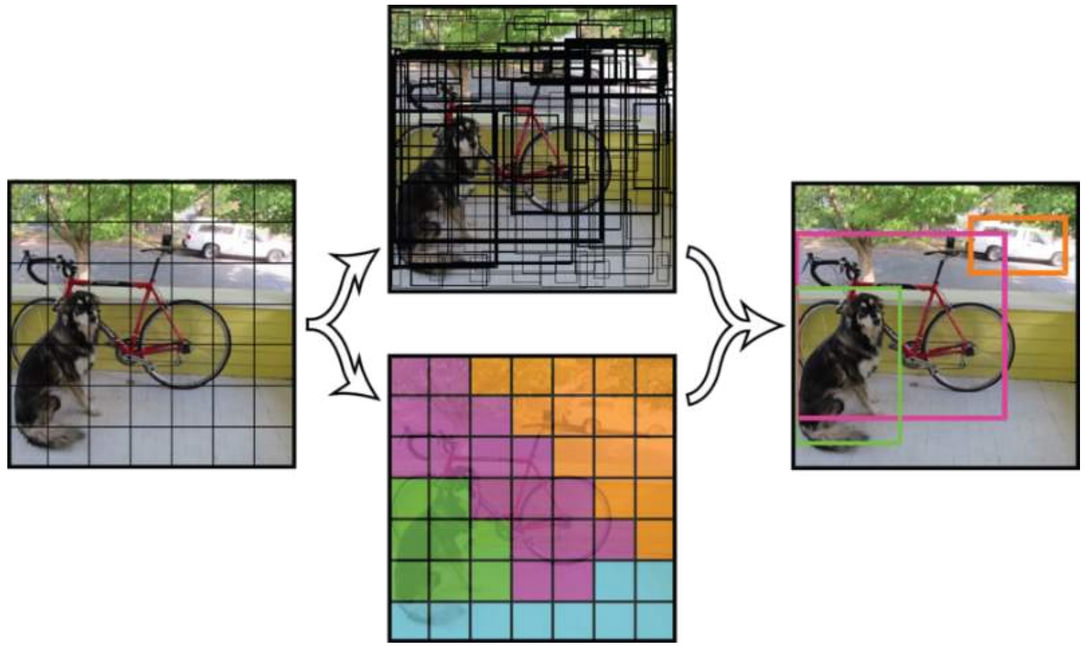
Appropriate for real-time object detection

Background

2) YOLO

Mark the locations where the objects are likely to exist with bounding boxes

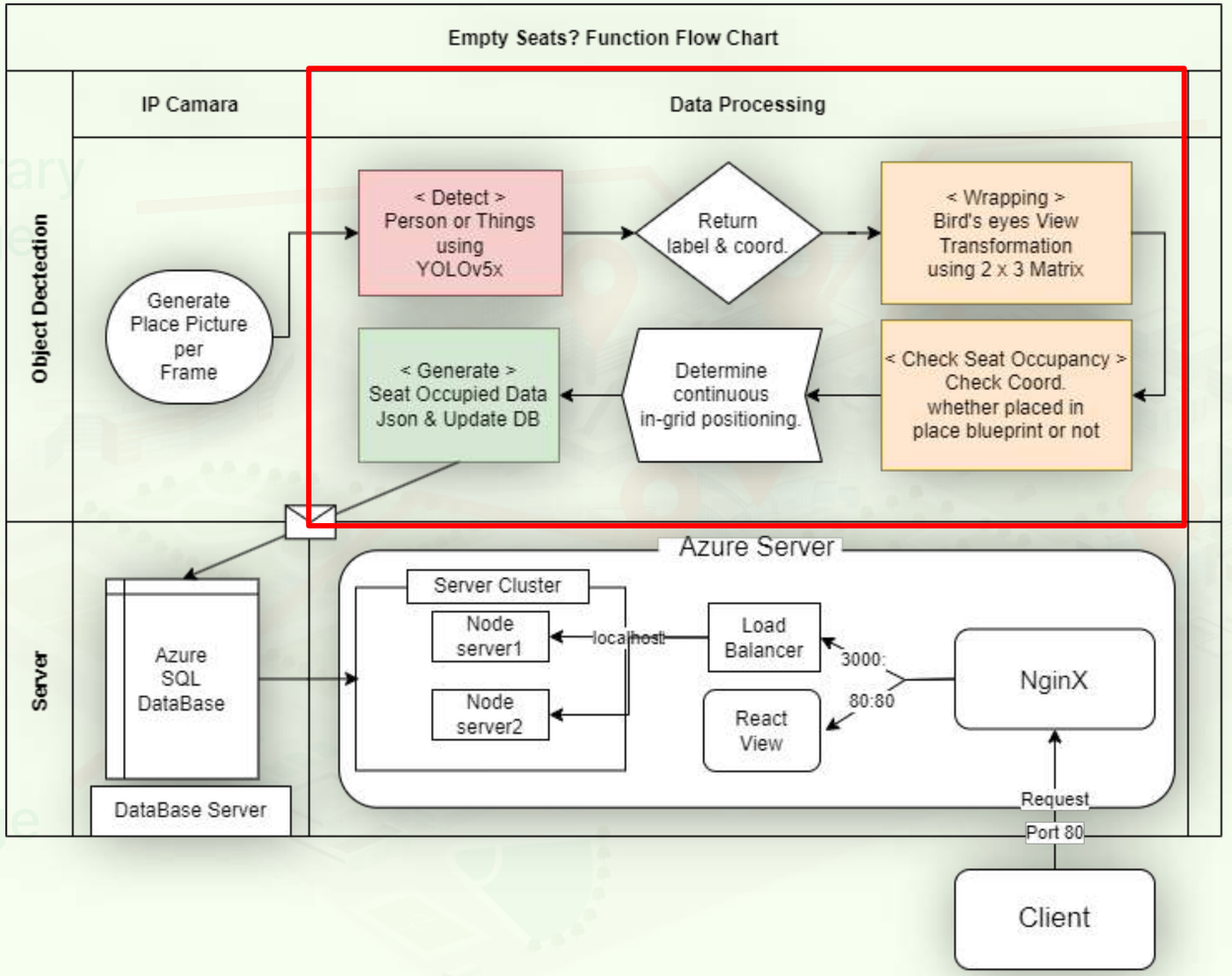
Divid the image into grids



Select the bounding boxes that are most likely to have an object

Indicates the class to which each grid belongs

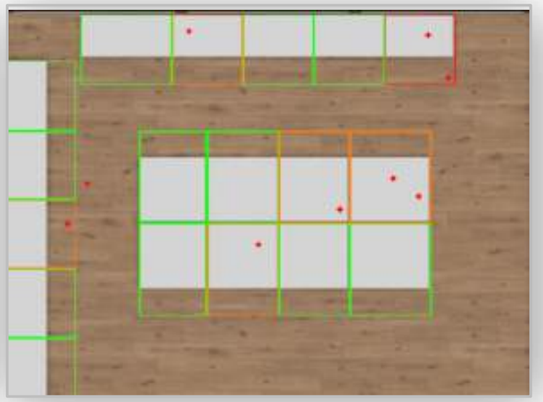
3. Implementation - Data Processing



Data Processing

Repeat every 20 frames (1 sec)

Detection → Transformation → Occupation → Update DB

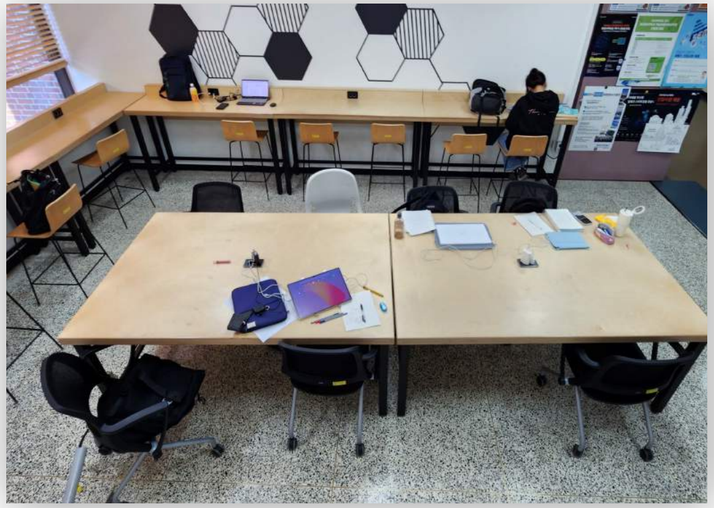


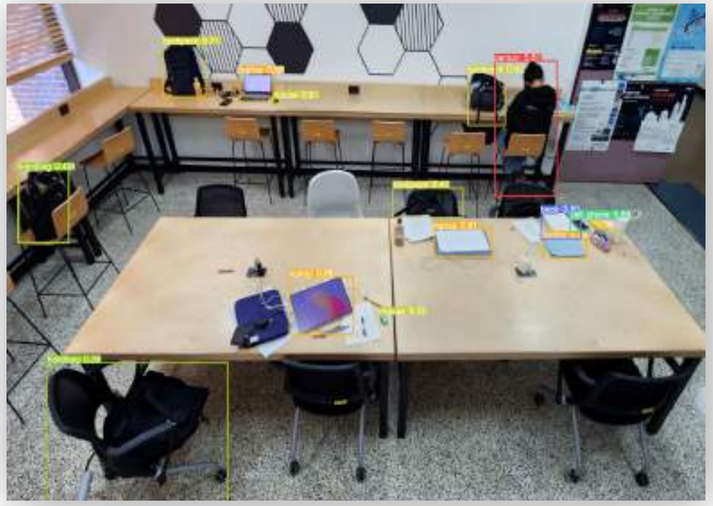

```
"parksangjo": {
  "seat1": 0, "seat2": 1, "seat3": 0,
  "seat4": 2, "seat5": 0, "seat6": 0,
  "seat7": 1, "seat8": 0,  "seat9": 2,
  "seat10": 0, "seat11": 0, "seat12": 1,
  "seat13": 0, "seat14": 2, "seat15": 1,
  "seat16": 0, "seat17": 0, "seat18": 2
}
```

PART 03 I Implementation

Data Processing

Detection → **Transformation** → **Occupation** → **Update DB**





YOLOv5x

```
label_map = {
    0: "person", 24: "backpack", 26: "handbag",
    62: "tv",  63: "laptop", 64: "mouse",
    65: "remote", 66: "keyboard",
    67: "cell phone", 73: "book"
}
```

```
[24, 0.055391434580008766, 0.5086705088615417, 0.10960118472576141, 0.2543352544307709],
[67, 0.8231905698776245, 0.4282711446285248, 0.022156573832035065, 0.0262743029743433],
[26, 0.1759231835603714, 0.39306357502937317, 0.060561299324035645, 0.14503414928913116]
[63, 0.6816838979721069, 0.46295323967933655, 0.07415066659450531, 0.05570152401924133],
[24, 0.7562776803970337, 0.38596951961517334, 0.0762186124920845, 0.06673672795295715],
[63, 0.5094534754753113, 0.580662131309503, 0.07769571989774704, 0.10930110514163971],
[24, 0.7100443243980408, 0.1886495053768158, 0.04697193577885628, 0.07882291078567505],
[0, 0.7590842247009277, 0.2320021092891693, 0.07474150508642197, 0.20861797034740448],
[63, 0.4311669170856476, 0.17682605981826782, 0.04224519804120064, 0.05202312022447586]
```

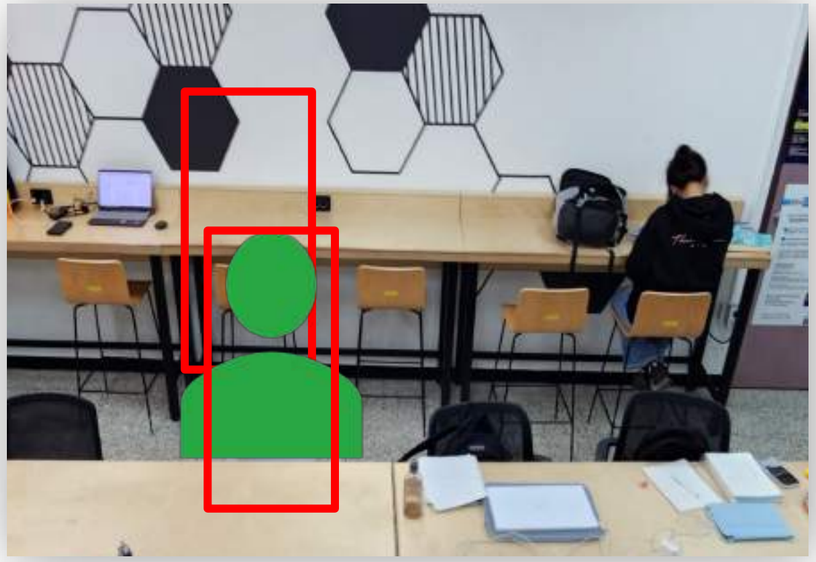
Label index & Coordinates
(x, y, width, height)

PART 03 I Implementation

Data Processing

Detection → **Transformation** → Occupation → Update DB

Challenge



Objects can be captured on **multiple** grids

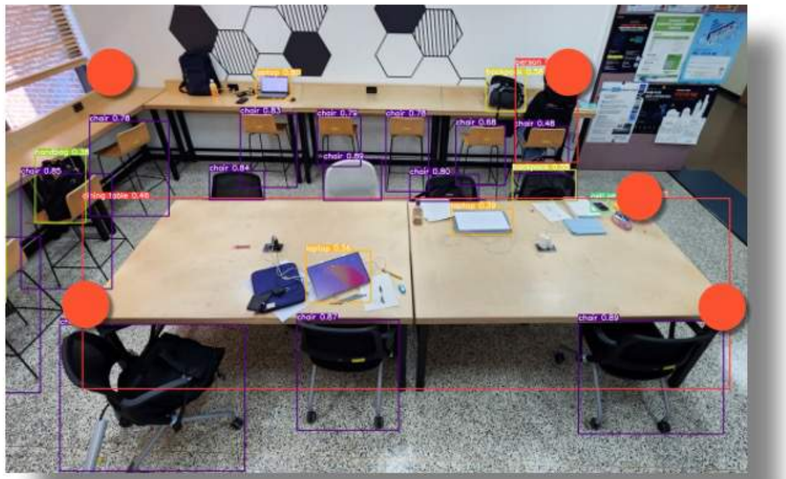
Solution

OpenCV Perspective transform

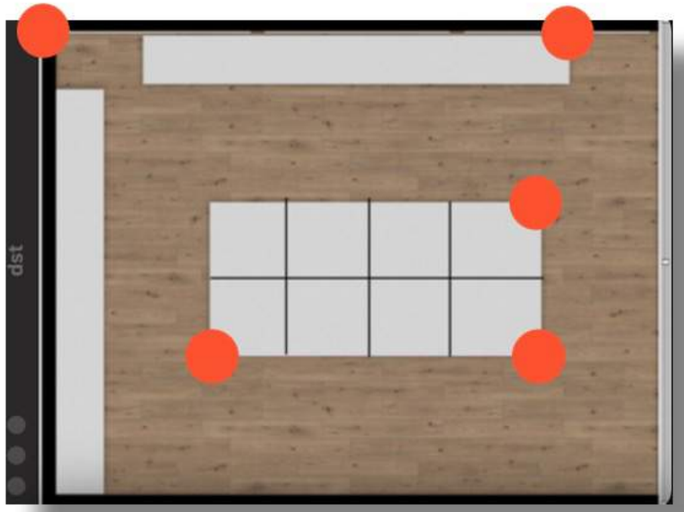
PART 03 I Implementation

Data Processing

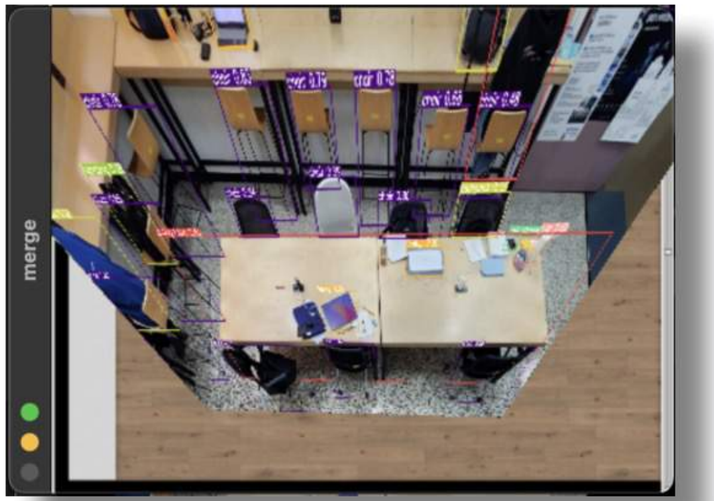
Detection → **Transformation** → Occupation → Update DB



Original image



2D space

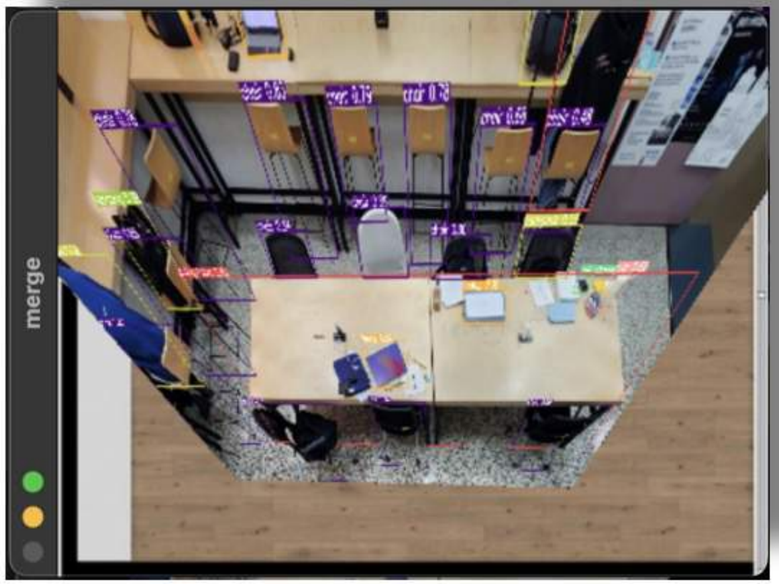


Transformed image

```
H1 = np.array([[5.60419376e-01, 9.71581689e-01, -3.79450113e+02],
               [-3.39450669e-02, 1.77097300e+00, -2.56173657e+02],
               [-1.75642895e-04, 4.53427216e-03, 1.00000000e+00]])
```

Data Processing

Detection → **Transformation** → Occupation → Update DB



Transformed image

Coordinates with labels only

PART 03 I Implementation

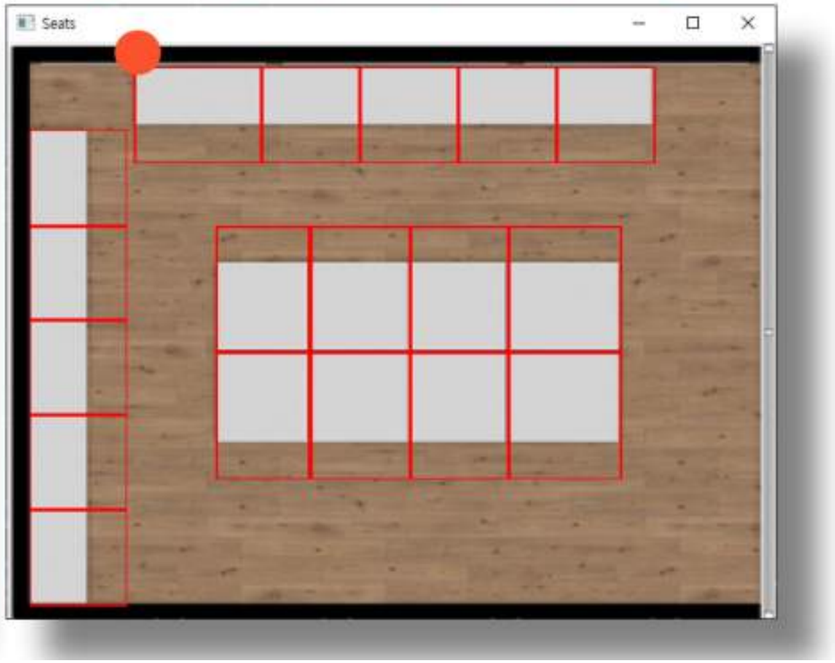
Data Processing

Detection → Transformation → **Occupation** → Update DB

1) Seat divison

```
# [x, y, width, height, direction of chair]
# direction = 1:Up, 2:Right, 3:Down, 4:Left
psj_lounge_desk = [
    [12, 262, 32, 53, 2],
    [12, 209, 32, 52, 2],
    [12, 156, 32, 52, 2],
    [12, 103, 32, 52, 2],
    [12, 49, 32, 53, 2],
    [68, 14, 68, 33, 3],
    [137, 14, 52, 33, 3],
    [190, 14, 52, 33, 3],
    [243, 14, 52, 33, 3],
    [296, 14, 52, 33, 3],
    [112, 174, 50, 50, 3],
    [112, 123, 50, 50, 1],
    [163, 174, 53, 50, 3],
    [163, 123, 53, 50, 1],
    [217, 174, 52, 50, 3],
    [217, 123, 52, 50, 1],
    [270, 174, 60, 50, 3],
    [270, 123, 60, 50, 1]
]
```

[x: 137, y: 14, w: 52, h: 33, dir: 3]



Added hard-coded desk grid for seat area designation

Data Processing

Detection → Transformation → **Occupation** → Update DB

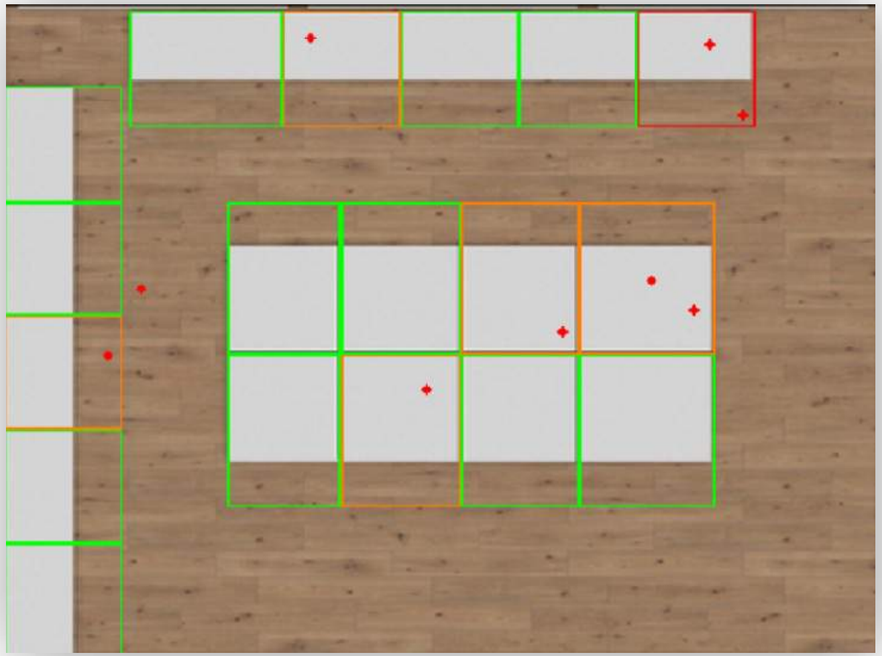
2) Seat occupancy indication

Grid-divided blueprint + Transformed coordinates & Labels

Data Processing

Detection → Transformation → **Occupation** → Update DB

2) Seat occupancy indication



- Occupied by person
- Occupied by object
- Available

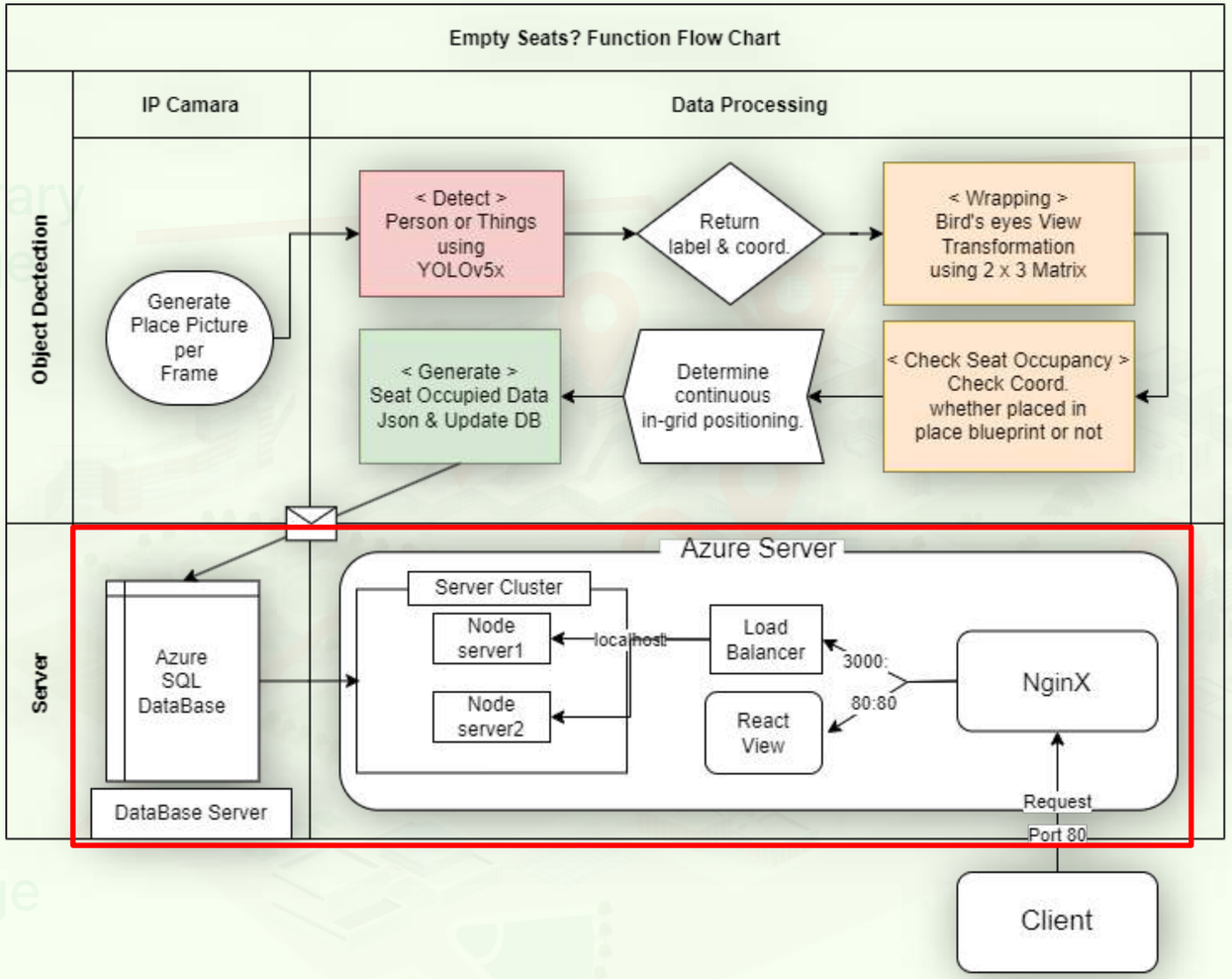
Data Processing

Detection → Transformation → Occupation → **Update DB**

```
"parksangjo": {
  "seat1": 0, "seat2": 1, "seat3": 0,
  "seat4": 2, "seat5": 0, "seat6": 0,
  "seat7": 1, "seat8": 0,  "seat9": 2,
  "seat10": 0, "seat11": 0, "seat12": 1,
  "seat13": 0, "seat14": 2, "seat15": 1,
  "seat16": 0, "seat17": 0, "seat18": 2
}
```

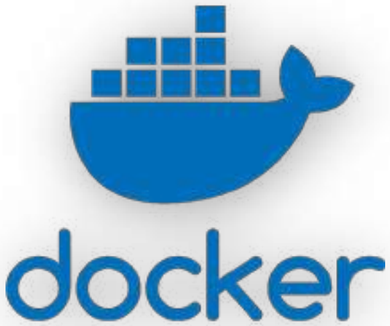
0: Available 1: Reserved 2: Occupied

3. Implementation - Backend



Backend

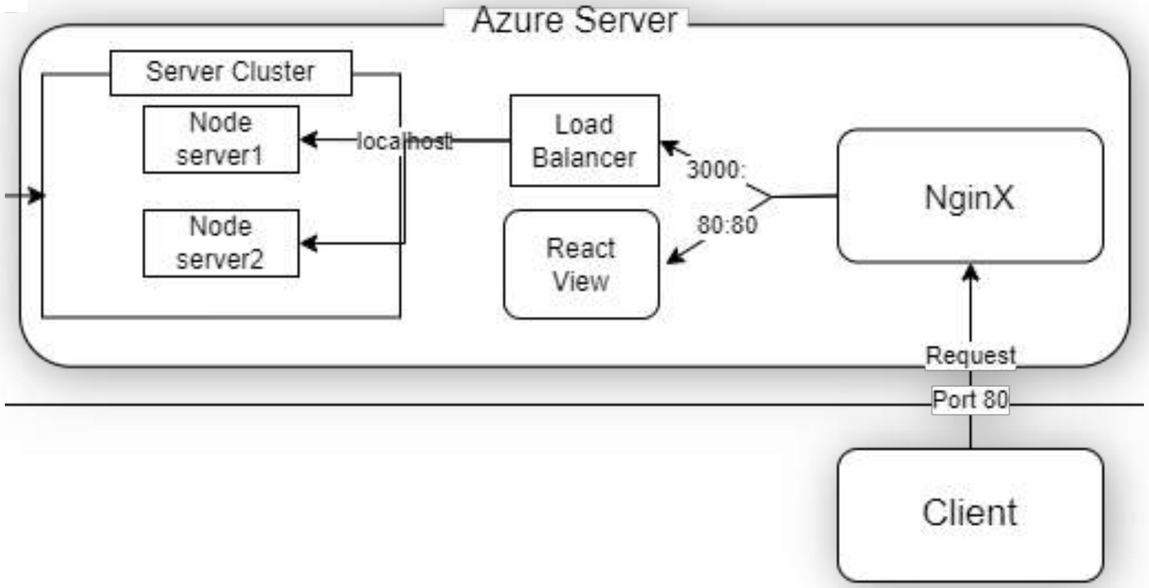
express



PART 03 I Implementation

Backend

Server Deploy Strategy



docker

•Sets up Node.js, manages dependencies, and enables container deployment.

•Establishes a multi-container system with Node.js server, React.js frontend, and Nginx for traffic management.

PART 03 I Implementation

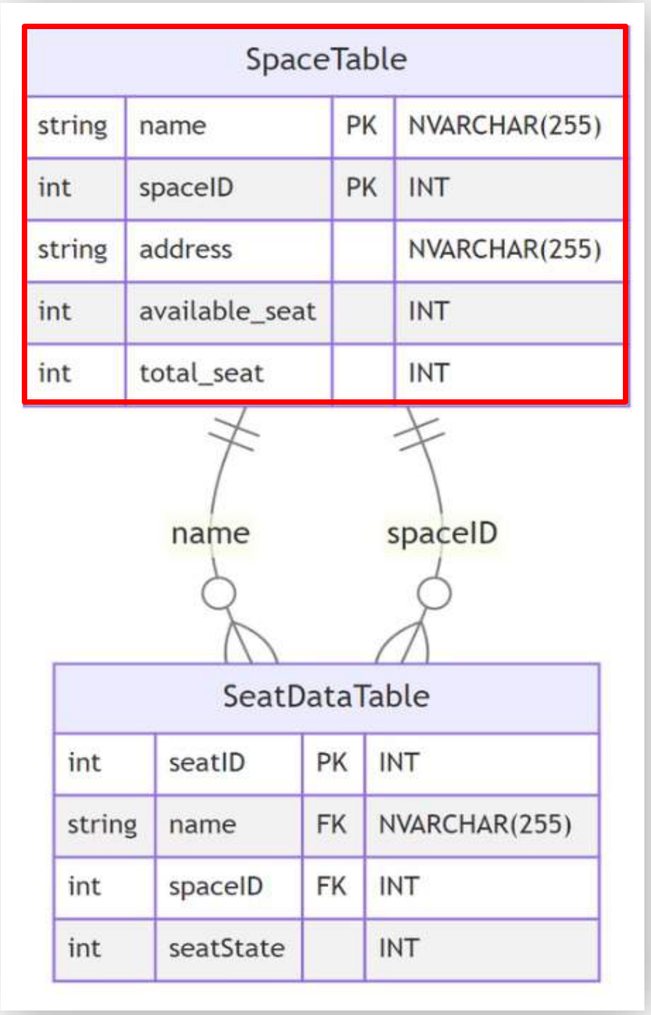
Database

1) SpaceTable

API: /data/getSpaces

name	spaceID	address	available seat	total seat
parksangjo	262201	Engineering No.2 Building 26, Floor 2	8	18
haedong	262202	Engineering No.1 Building 25, Floor 2	10	16

Represents different **spaces** in the application.



PART 03 I Implementation

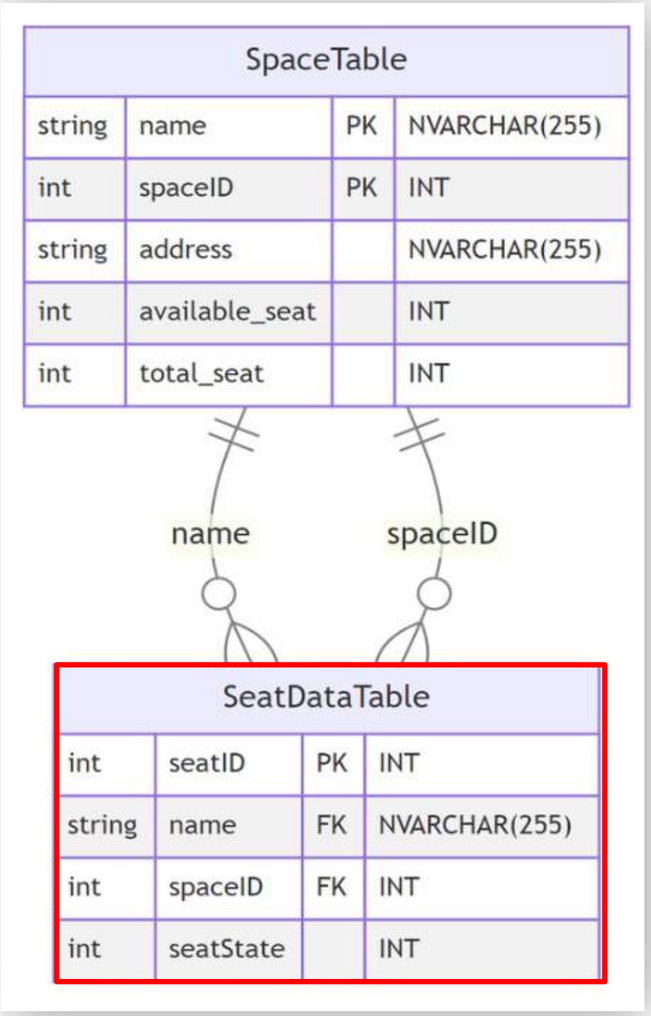
Database

2) SeatDataTable

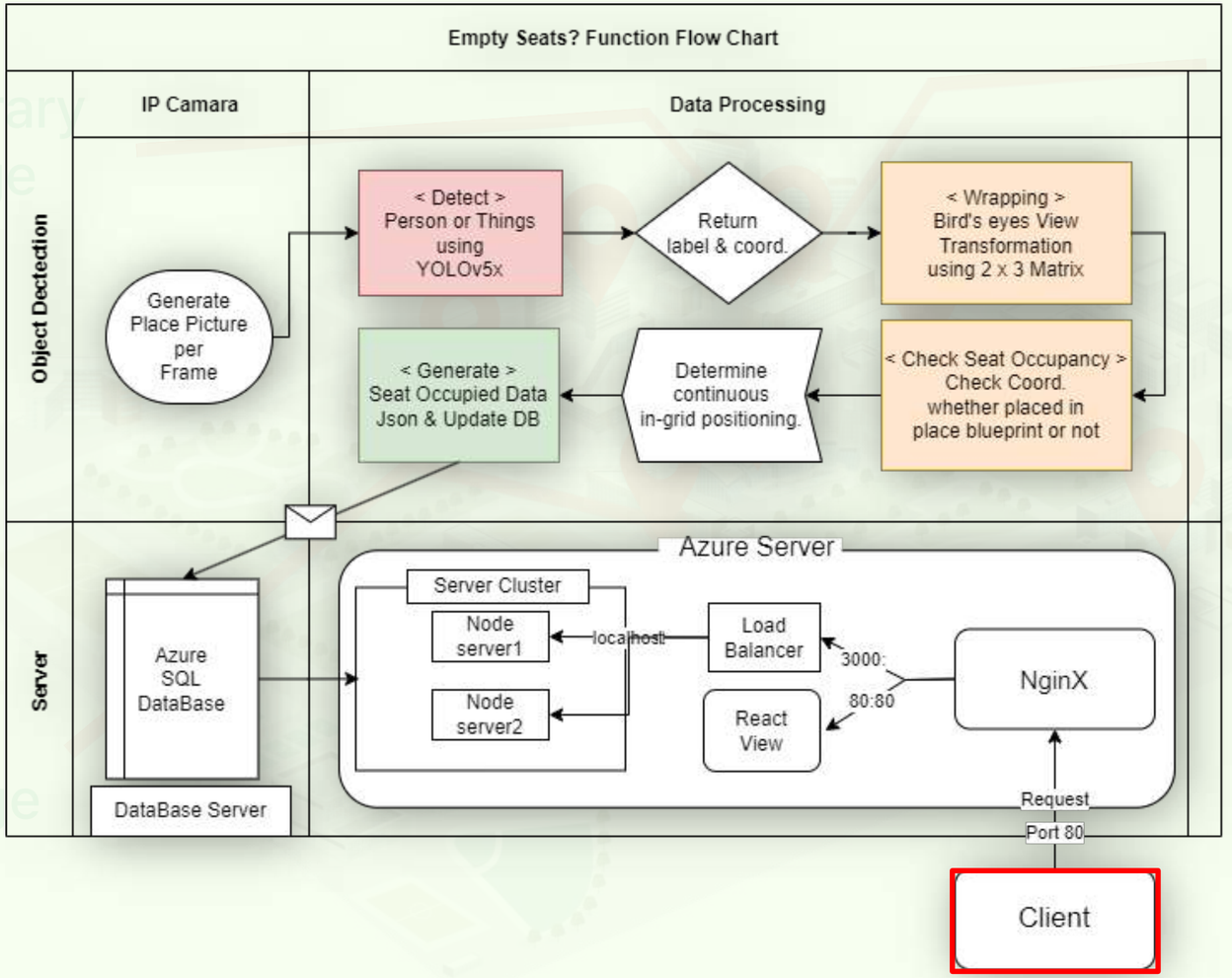
API: /data/getSeats

seatID	name	spaceID	seatState
26220101	parksangjo	262201	0
26220102	Parksangjo	262201	2
26220106	haedong	262202	1

Manages information about **individual seats** within spaces



3. Implementation - Frontend

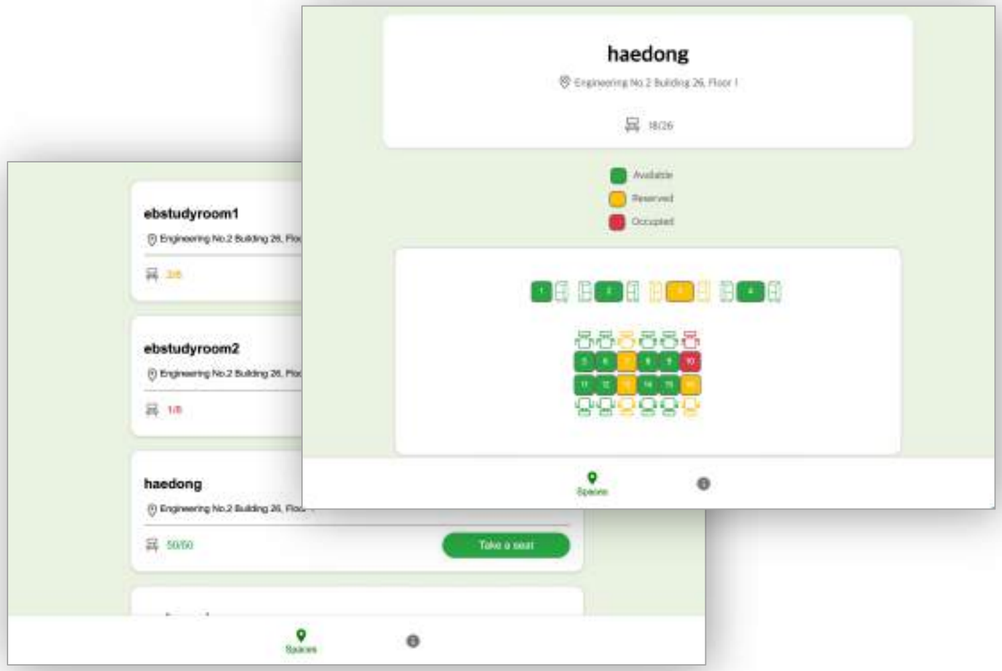


PART 03 I Implementation

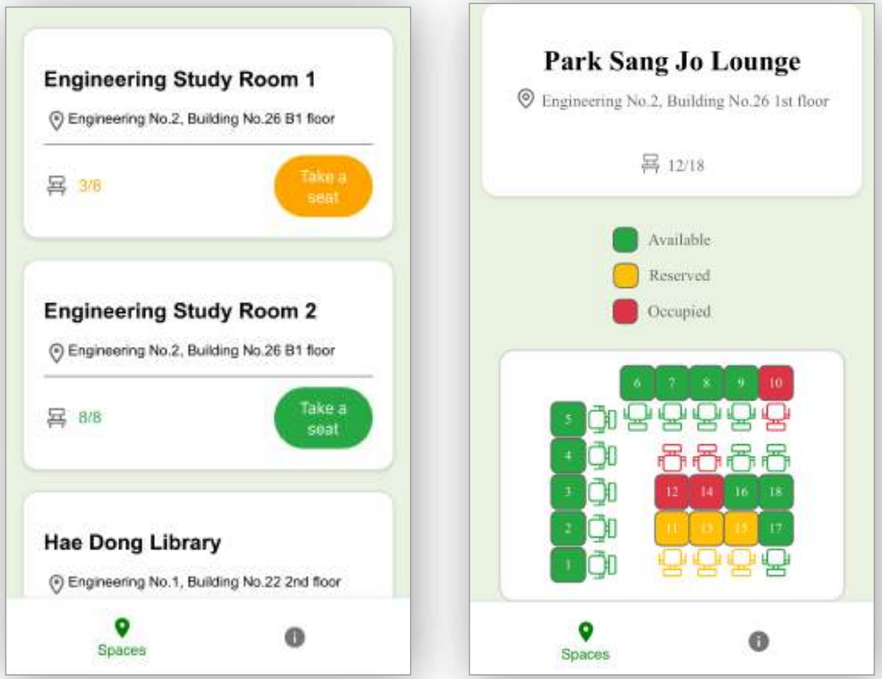
Frontend

React

A **web application** with a **responsive** design



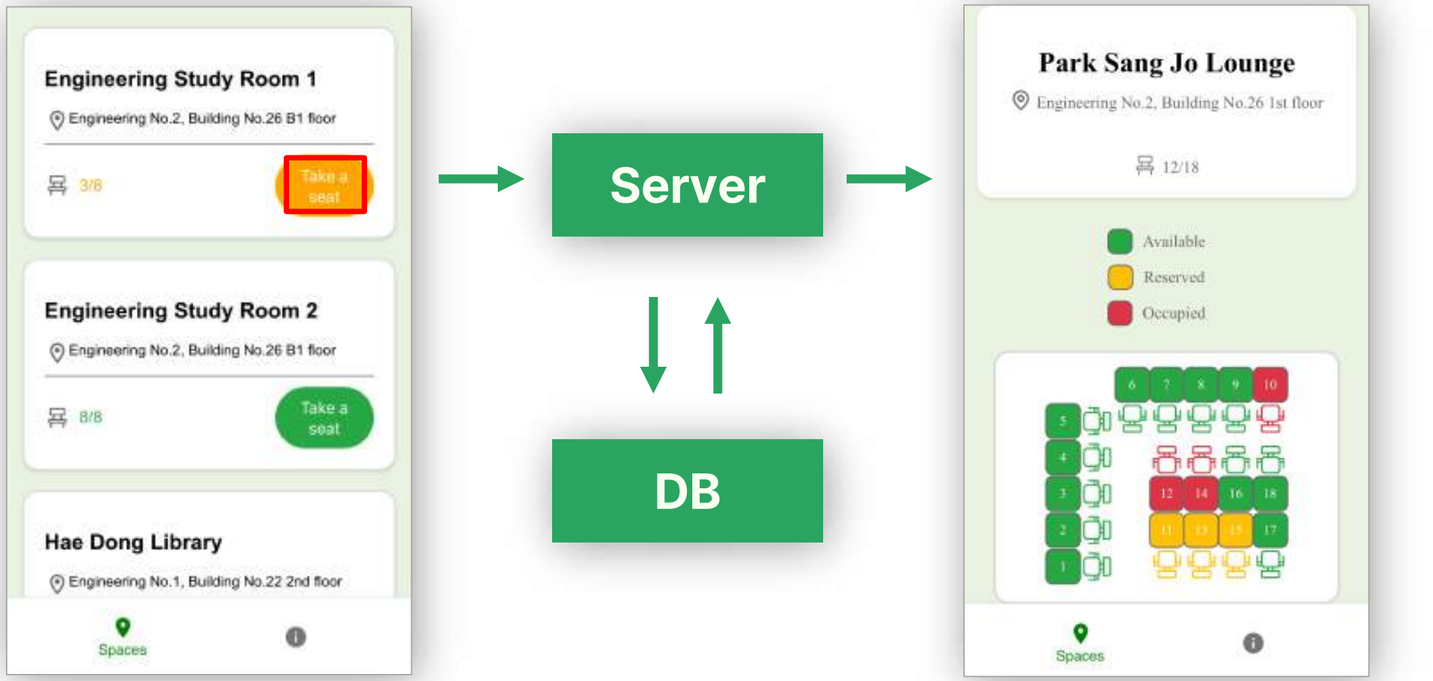
On desktop



On smartphone

PART 03 I Implementation

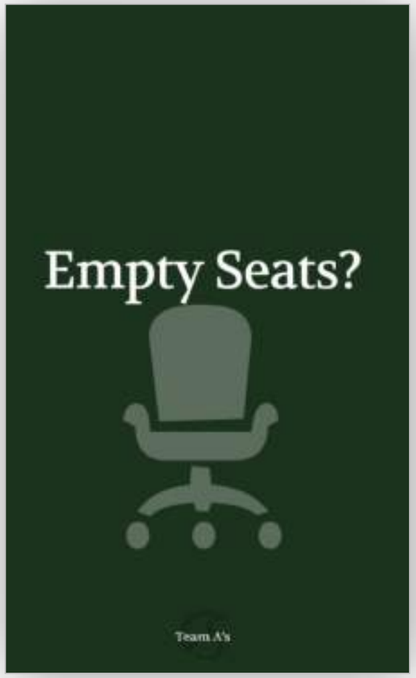
Frontend



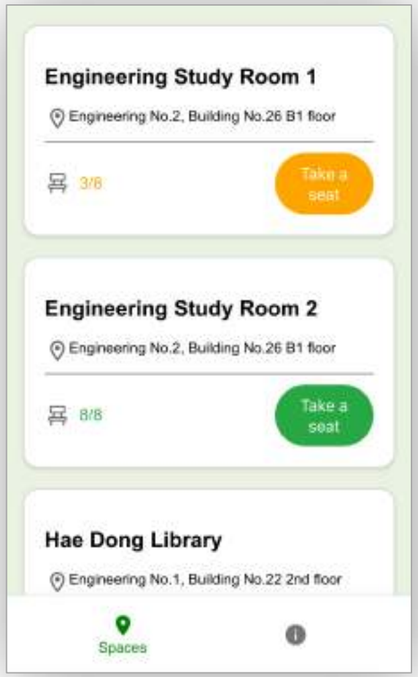

Samsung library
book lounge
Haedong
library
Eskara lounge

4. Design

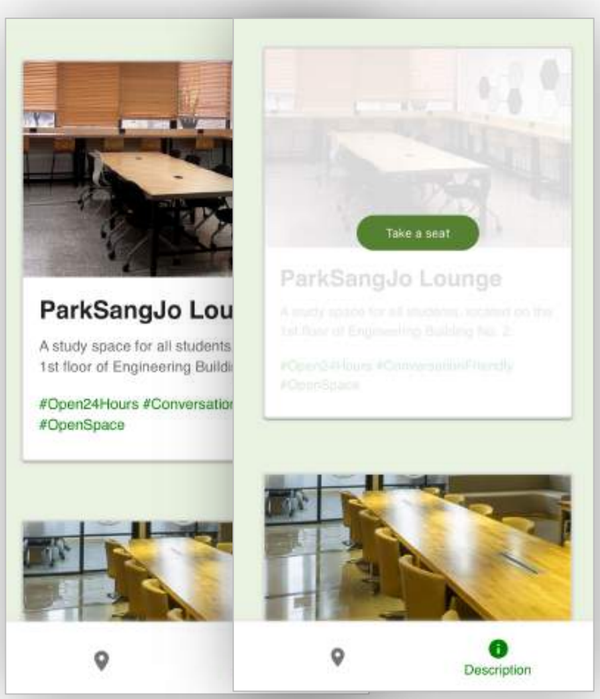
PART 04 I Design



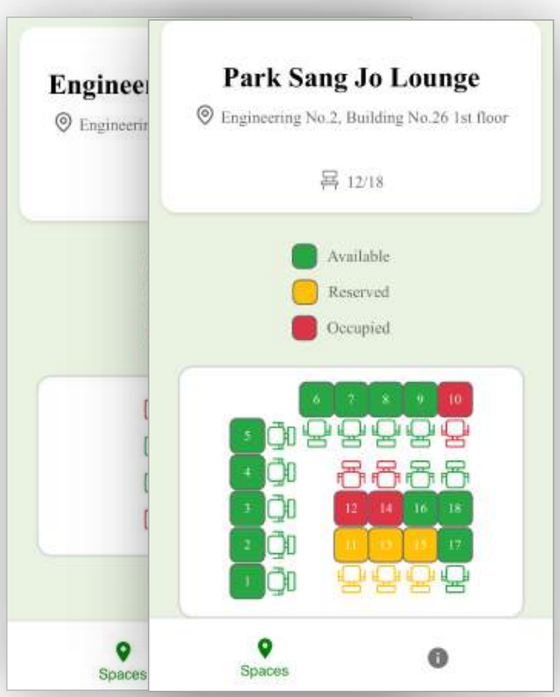
Splash



List



Description



Real-time status

PART 04 I Design

1) Space list

Location

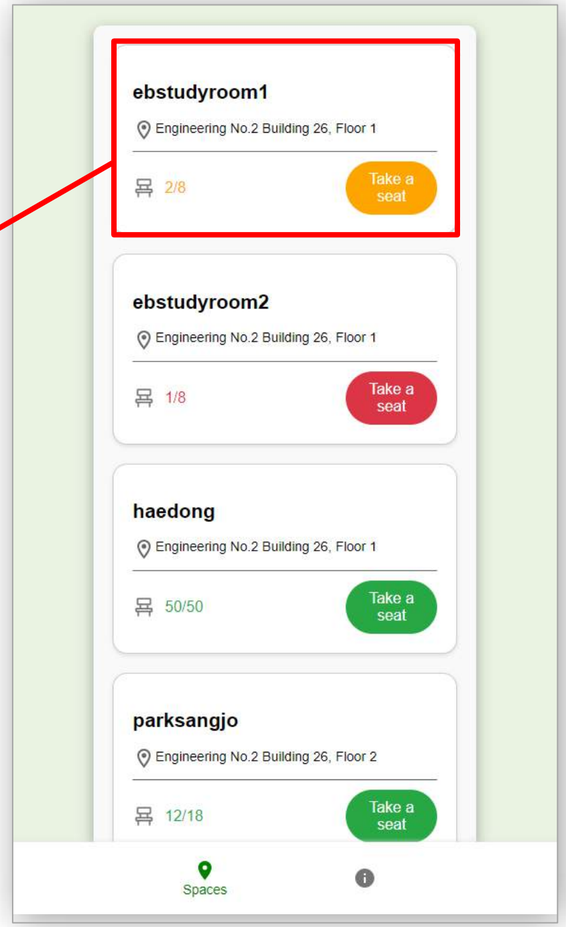
Available seats

Real-time status

Normal

Crowded(50%)

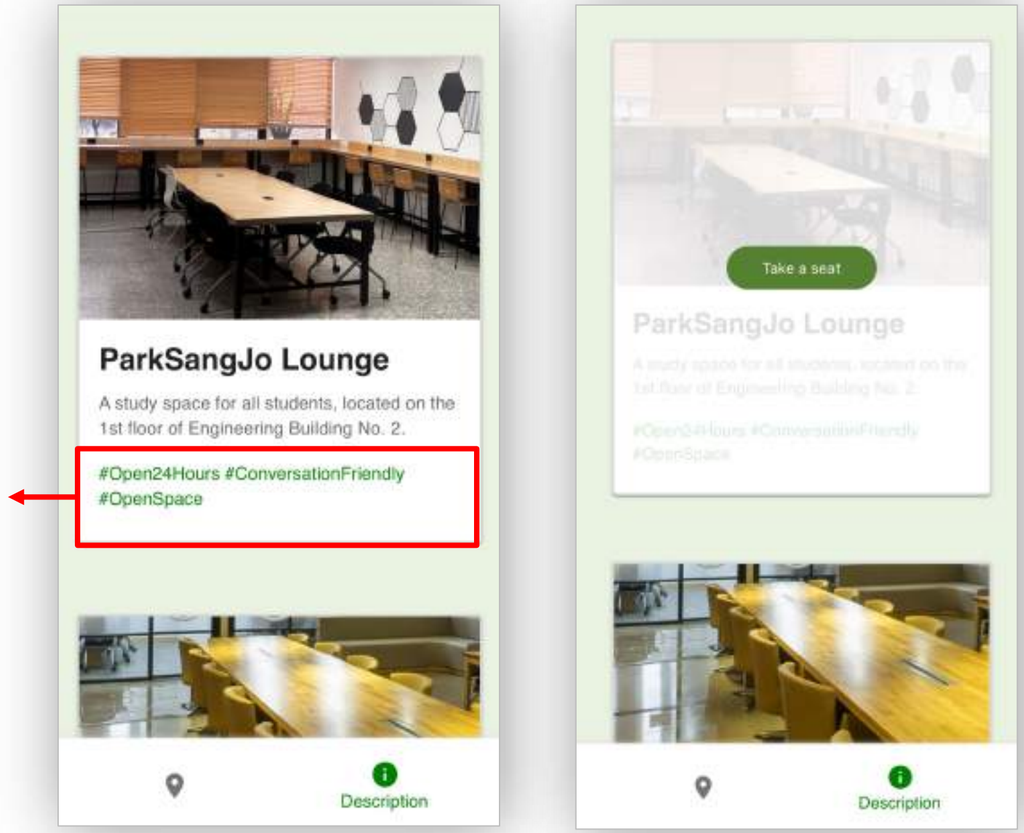
Very Crowded(80%)



PART 04 I Design

2) Description

Ambience

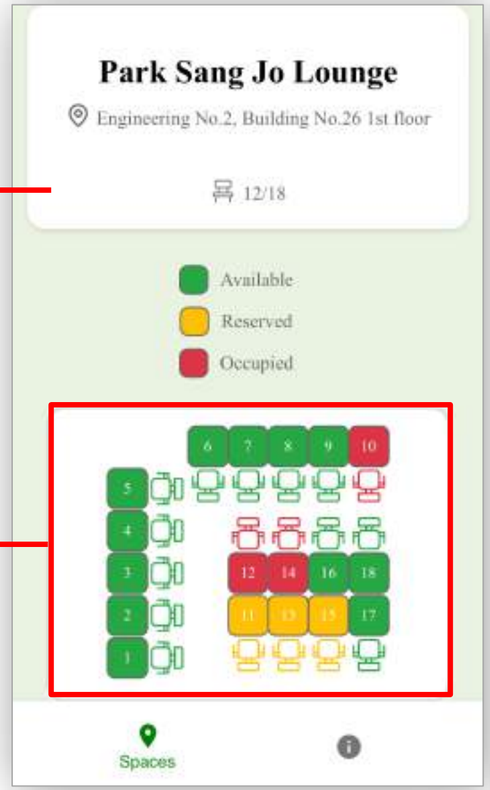


PART 04 I Design

3) Real-time status

Available seats

Seating charts for each space



PART 04 I Design

Demo

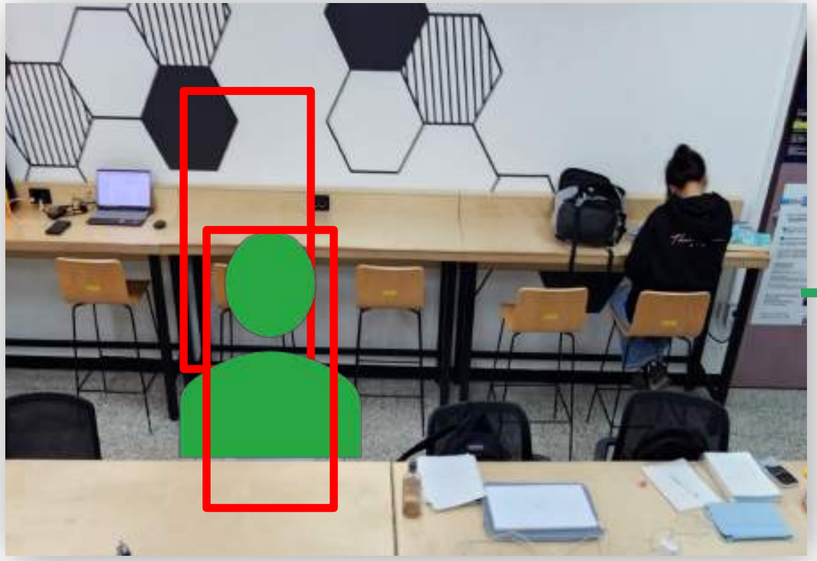
Go to see our website!

empty-seat.tech

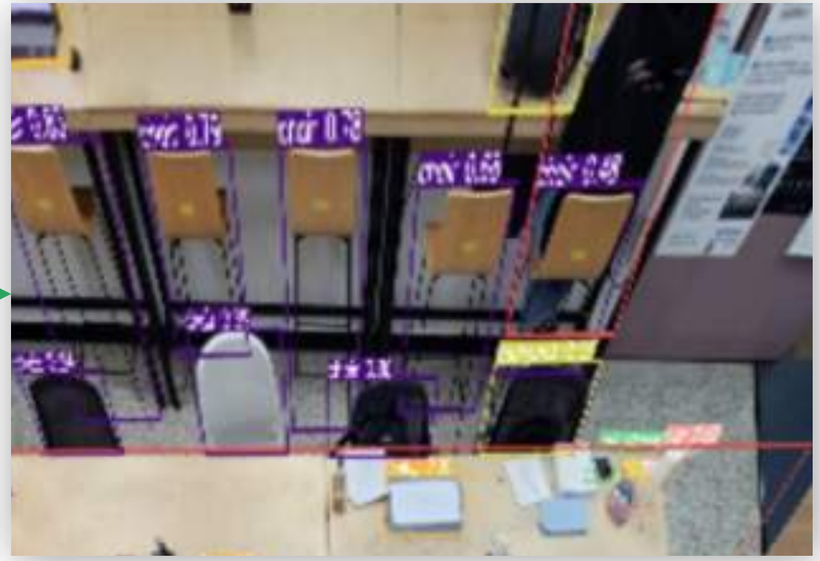
5. Conclusion

Challenges

1) Objects can be captured on **multiple** grids



Bird’s eye view perspective transform



Minimizing overlapping regions

Limitation

- The project was tested in a specific area within the school.
- Adjustment is required when there is a change in seating arrangement or camera angle.
- Expanding space can affect performance issues on your network or hardware.

Capstone Global Design

Final Presentation

THANK YOU

자리 있나요? Empty Seats?

TEAM A (A's) 김도엽 박재윤 우다연 최지민